O'REILLY®

Machine Learning at Enterprise Scale

How Real Practitioners Handle Six Common Challenges

Piero Cinquegrana & Matheen Raza

Machine Learning at Enterprise Scale

How Real Practitioners Handle Six Common Challenges

Piero Cinquegrana and Matheen Raza

Beijing • Boston • Farnham • Sebastopol • Tokyo O'REILLY®

Machine Learning at Enterprise Scale
by Piero Cinquegrana and Matheen Raza

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*http://oreilly.com*). For more information, contact our corporate/institutional sales department: 800-998-9938 or *corporate@oreilly.com*.

Acquisitions Editor: Rachel Roumeliotis
Developmental Editor: Nicole Tache
Production Editor: Nan Barber
Copyeditor: Octal Publishing, LLC
Proofreader: Nan Barber
Interior Designer: David Futato
Cover Designer: Karen Montgomery
Illustrator: Rebecca Demarest

June 2019: First Edition

Revision History for the First Edition
2019-05-22: First Release

978-1-492-05080-3

[LSI]

Table of Contents

Machine Learning at Enterprise Scale

Introduction

Machine learning. Seemingly everyone is doing it. And, if your company isn't investing in the skilled people and specialized tools needed to develop and deploy machine learning models, you're probably behind your competitors. Machine learning adoption was already high in 2017, at 58% according to a Deloitte survey of large enterprises, and it grew by five percentage points in 2018, to 63% of all respondents.

To get a sense of how fast businesses are adopting machine learning, IDC predicts that artificial intelligence (AI) spending (which encompasses machine learning) will grow to $52.2 billion by 2021. This represents a rather astounding compound annual growth rate (CAGR) of 46.2% within the 2016 to 2021 forecast period. This means that spending is increasing by nearly half again each year for five years.

Another sign that enterprise adoption of AI in general and machine learning in particular are increasing is that job titles specific to machine learning are already widely used at organizations with extensive experience in data science. According to O'Reilly's 2018 survey "The State of Machine Learning Adoption in the Enterprise", 81% of such businesses employ "data scientists." Thirty-nine percent of enterprises employ "machine learning engineers." And 20% employ "deep learning engineers."

These activities seem to be paying off. A recent Deloitte study found that the linkage between the successful application of machine learning and competitive advantage is growing (see Figure 1-1). Although only 11% of executives said that adopting AI is of "critical strategic importance" today, 56% believe it *will* be critical three years from now. Deloitte observes that this leaves "a very small window" for companies to hone their AI strategies and skills.

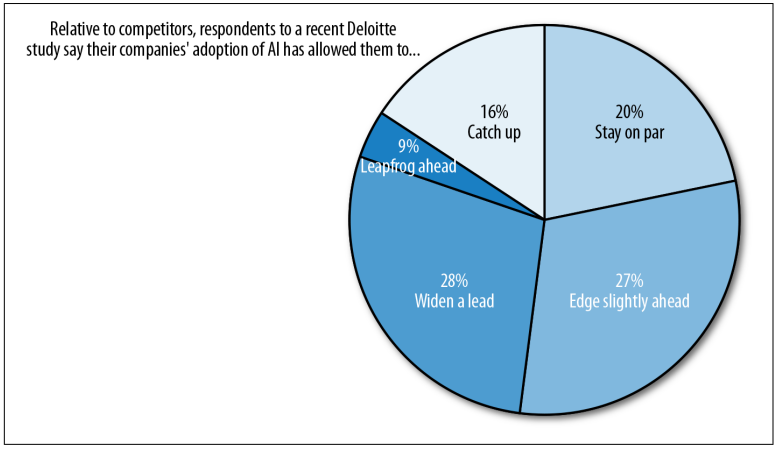

Figure 1-1. Many survey respondents say AI has helped their organizations keep up with, and even edge ahead of, the competition.

But particularly with machine learning, challenges abound. Another 2018 report, "State of Enterprise Machine Learning", surveyed more than 500 data science and machine learning professionals, and found the following:

- 38% of companies ran into problems scaling their models to the size they wanted. When asked to provide anecdotal answers, the data science professionals said this was due to a number of factors, including that DevOps and IT teams were not being allocated sufficient resources. Other reasons are that data scientists are being tasked with building infrastructure to put their models into production and the lack of existing infrastructure to support running machine learning models at the necessary scale.
- 30% of respondents reported challenges in supporting different programming languages and training frameworks. Because data scientists and machine learning engineers each prefer using

their own programming languages and training frameworks, this adds complexity because all these different tools must somehow cohere in a model that can be productized.

- 30% reported challenges managing models, such as keeping versioning clear, and being able to ensure explainability.

In addition, data wrangling remains a huge bottleneck. It's a truism in the industry that data scientists spend 80% of their time finding and managing data and only 20% building models. This general consensus has finally got some quantitative backing. According to the "State of Enterprise Machine Learning" study, despite the influx of investment in time, money, and personnel resources into machine learning, data scientists and machine learning professionals are spending too much time (more than 75%) on infrastructure, deployment and data engineering, and not nearly enough (less than 25%) on training and iterating models.

In our experience working with many enterprises to move their data science workloads to the cloud, we've identified six common challenges that data science teams need to address as they scale up their machine learning operations:

- Problem 1: reconciling disparate interfaces
- Problem 2: resolving environment dependencies
- Problem 3: ensuring close collaboration among all machine learning stakeholders
- Problem 4: building (or renting) adequate machine learning infrastructure
- Problem 5: scaling to meet machine learning requirements
- Problem 6: enabling smooth deployment of machine learning projects

This report explores these problems in depth and offers proven, practical advice for data scientists and machine learning engineers—advice that spans programming interfaces, workspaces, and data processing engines. In particular, we explore how managed cloud services platforms can offer a cost-effective, replicable solution to these problems.

We believe that there are valuable lessons to be learned from experts who have already taken this journey and built successful enterprise-

scale machine learning programs. For this report, we interviewed the following practitioners to present their unique perspectives and hard-earned advice on handling these common challenges.

Nakul Arora, Infosys

Nakul Arora is associate vice president, product management, and product marketing at Infosys, a global leader in technology services and consulting. Nakul manages the team that is responsible for the AI platform and business solutions at Infosys. The AI Platform, purpose-built to tackle IT and business problems, includes an automated machine learning workbench, a data platform, and an automation backbone. With over 200 engagements across various industries, the platform allows enterprises to build powerful horizontal and vertical-specific, custom experiences such as chatbots, cognitive search, contract analysis, and asset efficiency solutions, to name a few.

Patrick Hall, H2O

Patrick Hall is senior director for data science products at H2O, where he has acted as designer and data science lead for interpretability, transparency, and trustworthiness efforts for weak AI prediction engines, H2O Driverless AI. He advises some of the best commercial data science groups in the United States on advanced machine learning projects, and is coorganizer for the Washington, DC, Artificial Intelligence and Deep Learning meetup group. Patrick is also an adjunct professor at George Washington University in Washington, DC, creating lectures, assignments, and assessments for graduate data mining and machine learning courses.

Matt Harrison, MetaSnake

Matt Harrison is a consultant and corporate trainer at MetaSnake, a consulting firm that focuses on data science and, in particular, Python and related tools for machine learning. He has been using Python since 2000 across the domains of search, build management and testing, business intelligence, and storage. Matt also runs pycast.io, a screencasting service providing instruction on Python and data science.

Hussein Mehanna, formerly of Google Cloud

Hussein Mehanna was previously director of engineering for Google Cloud. Prior to joining Google, he was director of engineering at Snap and led a number of teams at Facebook, where he cofounded the Applied Machine Learning group. During his tenure at Facebook, he also founded a number of applied research teams, including AI platforms like FBLearner; Flow; and Caffe 2, the DeepText natural language understanding, a multilingual language-understanding platform used in Messenger, News Feed, and various other Facebook applications.

Joao Natali, Neustar

Joao Natali is the director of data science at Neustar, which makes an ID solution called One ID for large advertisers that includes consumers' online identities, offline identities, deterministic identities, and curated identities. Neustar generates data on its own systems, and also collects data from partners to create a tool with which its customers can interact, report, and perform simulations.

Jerry Overton, DXC Technology

Jerry Overton is a data scientist and technology fellow at DXC Technology, a global systems integrator. He specializes in industrializing analytics and AI for enterprise clients across all industries. Jerry is the author of the O'Reilly report "Going Pro in Data Science: What It Takes to Succeed as a Professional Data Scientist".

Sean Downes

Sean Downes trained in mathematical physics at Texas A&M University with an emphasis on supergravity and string phenomenology. He made the shift to industry by taking a position at a leading travel conglomerate in 2015, where he is now a senior data scientist. He presently works on assorted problems, particularly in platform and marketplace design and learning to rank initiatives.

Problem 1: Reconciling Disparate Interfaces

The first machine learning problem we identified is this: data scientists have certain frontend programming interfaces that they prefer to use. However, these interfaces are frequently disconnected from

the rest of the organization, especially if they're deployed on a laptop. The process of manually setting up interfaces to collaborate with peers, connect to execution engines, or connect to data stores can be extremely time consuming and painful for data scientists.

"At most companies, interfaces lead to very fragmented pipelines, with multiple players each having their own preferences, allegiances, and goals," says Nakul Arora, associate vice president, product management, and product marketing at Infosys.

Data scientists prefer to use certain tools such as RStudio for R programming or Jupyter Notebooks for Python programming. It isn't always easy for them to move to a different tool or technology. Data scientists' preferred tools and interfaces are different from—and largely incompatible with—those preferred by machine learning engineers, which are typically IntelliJ IDEA for Scala, Java, or C++. This problem is further complicated when data science work is being carried out on standalone systems like desktops or laptops.

To help each data science professional be as productive as possible, it's important to allow them to use their preferred interface while also providing a common workspace where everyone on the data team can collaborate.

What's the solution to this challenge? More and more data science professionals say that a cloud-first approach is essential for enterprise-scale data science and machine learning. Specifically, there is a need for cloud data platforms that have been developed to support the entire machine learning workflow. These cloud data platforms allow data engineers and data scientists to work and collaborate using the tools, languages, and data processing engines with which they are familiar.

According to a recent Deloitte study, 39% of companies say they prefer to use cloud-based services and platforms when deploying AI. In contrast, only 15% of companies say they prefer to use on-premises solutions. The popularity of cloud-based AI platforms is further confirmed by their annual global growth rate, which Deloitte estimates to be a "remarkable" 48.2%.

Data Scientists and Machine Learning Engineers: Different Roles, Different Tools

Before we get into the details of the first challenge, let's back up a bit and define the two primary personas in the data science life cycle that we will be talking about in this report: data scientists and machine learning engineers.

Data scientists

Data scientists apply scientific methods to data to build mathematical software models that generate insights or predictions, which then enable data-driven business decisions. Typically, data scientists are experts in statistical analysis and mathematical modeling, and proficient in programming languages such as R or Python.

Before they can even decide what model to build, data scientists need to explore the data, visualize it, run statistical analyses, and feature-engineer the data. It's very much a trial-and-error process, and notebooks play an important role. Notebooks are interactive tools that allow data scientists to quickly check the results of a piece of code or visualize data. They make the iterative nature of model building much easier.

Machine learning engineers

Machine learning engineers take the model—the code—that the data scientists have created and integrate it into a product. This product could be anything from a web application to a reporting dashboard, or an Internet of Things (IoT) device. Machine learning engineers typically have software engineering backgrounds and prefer to use programming languages such as Scala, Java, or C++. Sometimes, they might use Python, but that language tends not to perform as well for software applications when compared to Scala, Java, or C++.

The set of skills required of machine learning engineers when productizing data scientists' models is a bit different from those of the data scientists themselves. Although machine learning engineers are also expected to have a good understanding of statistics, they have strong backgrounds in software engineering. Machine learning engineering is a nascent role that hasn't been fully established across the industry. Regardless of whether they call them machine learning engineers, these people are an essential part of the data team.

In contrast to data scientists, who tend to use notebooks, machine learning engineers prefer using integrated development environments (IDEs) such as Eclipse and IntelliJ, that are specifically designed for building production-ready software applications.

The Three Interface Challenges Facing Data Professionals

Considering that these different roles employ different tools, three challenges arise:

- Collaborating effectively with peers
- Connecting to databases or data stores
- Connecting to large-scale execution engines like Presto, Hive, or Spark

In the rest of this section, we give you real-world anecdotes of how data scientists and other data professionals have overcome these challenges.

Collaborating with peers

How do you ensure seamless collaboration when your data scientists prefer one tool, and your machine learning engineers another? After all, peer-to-peer collaboration across data roles is essential to building a successful machine learning product. You have the data engineers preparing the data pipelines, and data scientists building the models who then hand them off to machine learning engineers to deploy those models to production. In some cases, the predictions from the machine learning model might be used by data analysts to make business-critical decisions. So, you have analysts in the mix, as well. Having the right interfaces in place to enable seamless collaboration is therefore essential. Unless you carefully plan, this can be an elusive goal.

Infosys' Arora believes the solution to this problem is simple: use a cloud platform that offers a plethora of interfaces. For example, a cloud platform might offer an interface into, say, both RStudio (which is where development or prototyping is often done by data scientists) and IntelliJ IDEA (which machine learning engineers prefer to use).

Such a platform must have *least-common-denominator capabilities* in terms of the number of interfaces, the types of interfaces, and the popularity of interfaces that it supports. In other words, interfaces should not be the limiter. "The more open and extensible your cloud framework, the better off you will be," says Arora. The extensibility of any platform is thus essential. Also keep in mind that interfaces evolve, and people's preferences evolve, as well. A platform that supports multiple interfaces keeps up with these changing customer and market dynamics.

What are these least common denominators (minimum viable set of capabilities) for a cloud data platform? Just what you might expect—RStudio, Jupyter Notebooks, and Anaconda.

Data scientists tend to not be as proficient as machine learning engineers when it comes to software programming skills. So, to collaborate effectively with their machine learning engineer peers, they might consider developing their programming skills—effectively becoming machine learning engineers. Otherwise, collaboration is difficult because they need to explain their code to machine learning engineers so that it can be refactored it into something that is production ready. This is what Sean Downes realized.

"Unfortunately, I think one thing severely lacking amongst data scientists is a technical maturity in programming skills," says Downes. He prefers to use IntelliJ whenever he can for writing batch jobs, JARs, tools, and functions when modeling code specifically because it's easier to collaborate with the machine learning and data engineers. "It's something I had to fight myself to do, but learning machine learning skills has sped up my own workflow dramatically as well as made it a lot easier for others on the team to see and understand what I'm doing," he says.

Connecting to existing databases or data stores

It doesn't matter which frontend interface you use. On the backend, you need an execution engine to connect to databases and data lakes, and to process data. Depending on the size of your data, you might need to use distributed processing engines such as Apache Spark or Presto.

DXC Technology's Jerry Overton says that although interfacing with existing databases is important, the bigger problem is how to get your model to connect to the rest of your digital ecosystem.

"How do you get it to interface with other vendors' apps, line-of-business applications, legacy systems, and existing enterprise data?" asks Overton. "That's really where the going gets difficult."

Best practices dictate that a data engineering team grabs all of the data from all the applications (Salesforce, Marketo, data warehouse systems, and many others) and dumps it into a data lake. Then, the interfaces connect to this data and query, train, or process it through a big data engine such as Apache Spark, Hive, or Presto.

Although Overton identifies a slightly different interface *problem* than his peers, he recommends the same type of solution.

"You need a cloud-driven platform to do this—a Platform-as-a-Service," he says, adding that a number of existing solutions not only host an environment for you, they also make it easy for workstations and laptops to connect to it. "And, you can even collaborate with other developers—often in real time," he says.

We can also consider dashboards as a type of interface, and data teams need to configure and connect them to data processing engines. When it comes to dashboards and displaying the results of models, Infosys' Arora stresses that the world has moved past Tableau. "There's Qlik, there's Microsoft Power BI, and, quite frankly, there are three different clouds, or, perhaps we should call them 'walled gardens': Microsoft Azure, AWS, and Google Cloud," says Arora. "They each have AI tools for visualizations within those ecosystems. So being able to interface with them also is a key baseline capability."

Joao Natali at Neustar says he hardly ever uses things that are prepackaged for visualization. "We never use Tableau or anything like that, and very little Excel. We pretty much focus on notebooks and writing code for everything we have to do," he says.

Connecting to large-scale execution engines

Different people prefer to use different languages, such as Scala, R, or Python, when connecting to large-scale execution engineers.

At Neustar, data scientists prefer using the typical Python data science stack. "Sometimes we use R, sometimes we use Scala," says Natali. If there is a dependency from engineering on interfacing, there's always the Java Virtual Machine (JVM), he says. "But if we

can choose without restrictions, we are essentially a Python-based shop."

Neustar's machine learning engineers, on the other hand, prefer Java to Scala or other tools when connecting to the large-scale execution engines. The language you choose will determine the interface you use. For example, Scala users typically prefer IntelliJ or Eclipse.

"Scala has a lot of nice things, but it also is a very free language in the sense that two people can solve a problem, and their code will in no way resemble each other's," says Natali. As a result, teams that use Scala invest a lot of time in creating code standards.

"We don't have time for that," says Natali. "That's why Python is a good solution for us."

In fact, most data science practitioners agree that Python is ideal because it is more production ready than R. It also has more general-purpose libraries in areas other than data science. And, its structure is more robust, with different classes and types.

In summary, when it comes to interfaces, two things are important to consider:

- First, depending on their role and technical expertise, members of the data team might prefer different interfaces. Machine learning engineers might prefer programming IDEs such as Eclipse or IntelliJ, and data scientists might prefer RStudio or Jupyter Notebooks.
- Second, interfaces are a way to connect to and process data. Connecting the interface to the engine is a time-consuming process. A cloud-based data platform that preconfigures the engine and interfaces provides consistency across the organization and allows data scientists to focus on their core task of building models.

Problem 2: Resolving Environment Dependencies

Many data-science challenges arise when data or code moves between different environments. In computer science, an environment denotes the specific dependencies between your operating sys-

tem version, your programming language, and the libraries that you are using to build your models.

Two challenges, in particular, have been identified by our experts: DevOps bottlenecks and code-portability issues.

Setting Up a Machine Learning Environment and Avoiding DevOps Bottlenecks

All too often, data scientists are expected to stand up their environments themselves, or wait until DevOps professionals have the time to spin up an environment for them. Either way, challenges arise—either the data scientist doesn't understand the production environment sufficiently to avoid migration issues, or the DevOps team is too busy and becomes a bottleneck in the process.

According to Infosys' Arora, part of the challenge is that data scientists don't take the models into production. So, when they set up their environments, they might be unaware of what's coming next.

"Really, the only way you minimize the pain or minimize the redos or doubling up of efforts, is if you provide an environment that they can use to model from development, to test, to production," Arora says. However, although that is ideal, it is very difficult—probably impossible—to find that rare person who is an expert in math, statistics, computer science, and cloud DevOps.

That's why, in most companies, the IT department usually sets up a development environment for the data scientists. And there is some level of testing that they're able to do after models are built.

"But, when it comes to the production environment, it's a completely different group that handles that," says Arora, who believes there is a need for a bridge across the development and production phases.

The ideal solution would be for the data scientist to work in a much more production-ready environment so that they're able to more seamlessly deploy their code into production. Compared to a traditional siloed environment in which you need to hand off the code to somebody else who has access to the production environment, data scientists should be able to experiment, test, and move code into production themselves. This is what a cloud-native data platform enables.

Natali of Neustar says his firm typically goes for the "easy" solution when it comes to environments. "If we know that a particular piece of code is going to run in a cluster, or we can define a cluster as the configuration that we need, then we can build code specifically for that cluster," he says. "And we can easily set up the cluster, run, and test it, and move it into production without hassles."

His organization has a team called *CloudOps* that sets all this up. Similar to what other organizations call DevOps, CloudOps is responsible for handling provisioning infrastructure across multiple platforms: on-premises, private cloud, and public cloud.

Solving the Code Portability Problem

As mentioned earlier, we find many data science teams still work off laptops or local systems. The challenge with building machine learning models on a laptop is code portability between environments. As you move code from prototype to production, all of the dependencies need to be transferred or duplicated in the production environment. And that could be a major effort that might require multiple iterations.

Suppose that you develop a model on your laptop and then try to deploy it on a server or a cloud virtual machine (VM). The code is likely to break because you are probably using a different library version or even the same version number but for Mac or Windows, whereas you are deploying it on Linux. It could take significant effort to duplicate all of the package dependencies and continue to maintain them in both the local development and production environments. This can also present discrepancies and complications during QA processes, thereby unnecessarily extending time to production.

This is a huge problem that both data scientists and machine learning engineers have to deal with.

In an ideal world, a development platform should be able to support any and all frameworks that are available. You should be able to import libraries or import models that have been built in, for instance, the Facebook world or Google world, or Microsoft world, or any other frameworks that are out there. But that is rarely the case.

Code portability is a big enough pain point that Natali's company, Neustar, simply avoids it. "We no longer perform computation or analyses on our laptops," says Natali. "We do everything in the cloud. We could even depend only on Chromebooks because our laptops are just a way to connect to a remote machine that does all the number crunching."

These remote machines are set up in a way that they can access cluster systems in master roles and are easily reproducible. In effect, Neustar puts development as close as possible to where things run. "It doesn't work perfectly all the time, but it's the best solution we've found," Natali says.

Being able to reproduce a production environment and being able to define and control it tends in most cases to be good enough, "and tends to be simpler than dealing with containers," says Natali.

Another challenge that many businesses face is that they have hybrid environments, with some data existing on-premises for legal reasons, but a lot of data also stored in the cloud. In such cases, they don't want to write code for their on-premises solution and then waste time writing it again for the cloud. Using containers for deployment can help alleviate the need for rewriting code for different environments. However, with larger datasets, containers can become cumbersome to manage. You need skilled DevOps engineers who know how to efficiently deploy containers into infrastructure; this is an important consideration when deciding whether to work with containers.

Hussein Mehanna, formerly of Google Cloud, points to Google's Kubeflow solution as one option. Based on Kubernetes, it allows data scientists to create a unified abstraction in which they write code once, and it runs everywhere.

Similarly, Docker was a big help in the deployment world, says H2O's Patrick Hall. Four or five years ago, before Docker became prevalent, it was very difficult to get things in R and Python deployed. "R and Python are not, in my opinion, production-grade tools," says Hall.

He points out that when you're starting a project, you need to consider how you are going to take the thousands of lines of Python code, and move them into a public-facing, secure, and high-

availability environment that's actually going to make or save your company money.

Docker helped H2O with this. It allowed Hall to encapsulate his development environment, and move it to his production environment.

However, there are corner cases, such as real-time, mission-critical workloads, for which this strategy might not work, Hall warns. To overcome this issue, he uses a machine learning library that at the end of training is able to generate a piece of code in a deployment-friendly language or format that he can immediately transfer into his production environment.

The way H2O works is this: when you are finished training your model using H2O, you get a piece of Java code. You then take that Java code—with no human translation required—and move it into your production environment. It doesn't matter whether it is a more traditional database that can run Java, something in the Hadoop or Spark environment that can run Java, or a custom Java application. It just works.

But although reconciling differences in environments is a challenge, another issue tops it. That is, the models you are creating are rarely big enough or complicated enough to give the insights necessary to make a real difference to your business.

Models are usually part of a larger ecosystem of models. The output of one feed into the input of another, according to Overton. "You need some sort of standardized approach to discovering new models in your environment, hooking them together, making sure that they're sharing information back and forth, and that it is being done in a standard way."

Overton calls this the *utility approach*. "What you want to do is put down a utility infrastructure so that you can plug things in and pull things out while ensuring there's interoperability," he says.

In this section, our experts identified two challenges related to reconciling different environments: DevOps bottlenecks and code-portability issues. DevOps bottlenecks occur in organizations that have not yet deployed a self-service model.

Code portability remains a challenge for even the most sophisticated organizations. Data scientists and machine learning engineers con-

tinue to struggle with it. This issue can be solved by adopting a cloud-native platform for data science that is able to support all available frameworks. You should be able to import models and libraries built into any framework so that code transfers smoothly from one environment to another without breaking. A data platform like Qubole, which is flexible to users' changing requirements, can help companies keep up with the latest open source data engines.

Problem 3: Ensuring Close Collaboration Among All Machine Learning Stakeholders

Interfaces, environments, and machine learning models are just some pieces of the larger machine learning puzzle for enterprises. Data scientists, data engineers, machine learning engineers, data analysts, and citizen data scientists all need to collaborate to deliver machine learning–based insights for making wise business decisions.

It's a team sport.

As we've discussed, data scientists must collaborate closely with engineers to create machine learning products. Data engineers help data scientists build production extract, transform, and load (ETL) pipelines. And machine learning engineers deploy data scientists' code. They all must work seamlessly together.

Often, these collaborative efforts happen via email and GitHub. However, this is inefficient because data scientists cannot see the work that others are doing until it is committed to a repository.

Having better tools that allow continuous collaboration and governed searches of the code, data, and metadata of peers leads to much more optimal outcomes.

Attitudes are important, too.

"You basically don't allow primadonna behavior where a data scientist decides that all he or she does is write algorithms and how the model runs and where it runs is not really their problem," says Overton. "Or a data engineer who says, 'I deal with the infrastructure and I don't really know what these algorithms are for, and I don't want to figure it out.'"

The first step in solving the collaboration problem is making sure that you have a culture in which that kind of behavior is not tolerated.

In short, when productizing machine learning models, data scientists are just one piece of the puzzle, and collaboration is critical. Collaboration means sharing information. It means data scientists sharing code with one another. It means data engineers, data scientists, and machine learning engineers sharing metadata about data sources. And it means enabling model auditability and explainability so that data scientists can collaborate with business stakeholders like citizen scientists and executives.

A cloud-native data platform can help enable all of these different types of collaboration. By allowing users to see results from code and iterate over the code before committing to a code repository, you prevent code and data from becoming stale. In addition, a cloud data platform also provides a metadata store and data catalog that can be shared across all use cases.

Collaboration Between Data Scientists

First and foremost, data scientists need to collaborate with one another, peer to peer. When your data scientists are *not* doing this, valuable institutional knowledge can be lost if it's locked in the heads of one or two persons.

"Our customers have reported that among their teams, data scientists aren't always aware of each other's work, and sometimes reinvent the wheel," says Mehanna, who adds that collaborating on code through email and GitHub doesn't really work. At Natali's Neustar, data science team members collaborate in two ways. One is through Git-shared repositories. "We start a repository, and everybody can look at it," says Natali. "That way, everyone knows what we're doing."

As a second way, some vendors are trying to address the code sharing problem by creating concept data boxes (also called model or architecture boxes), which are sets of prebuilt architectures or models that you can apply to a new dataset, or which can get you started for building a new model.

With these kinds of tools, if you're trying to build a recommendation model or engine, "you can pull up other models or architectures that might be applicable to the task at hand," says Arora.

Even though Harrison, of MetaSnake, champions the use of Jupyter Notebooks, he admits that there are some drawbacks as far as collaboration is concerned. His experience underscores the fact that these tools were not really meant for collaboration.

As an example of how such tools are less-than-perfect is the fact that Jupyter—the most popular notebook for data scientists—uses JSON "under the covers" as a storage format. If you have images that are embedded in the JSON files, your files can become very big, and standard diffing tools have problems doing diffs and deltas. "So it makes collaboration a bit more difficult," says Harrison.

Overton points out that because many of the tools currently in use aren't really collaboration platforms, the best way to get your data scientists collaborating is simply to sit them down together and make them talk, says Overton. You see that kind of thing built in to practices like stand-up meetings and mob programming. "There's all kinds of standard practices that are aimed at solving this collaboration problem between your data specialists," he says.

Collaboration with Data, Machine Learning, and Software Engineers

Data scientists also need to collaborate with data engineers, machine learning engineers, and software engineers. Much of this collaboration revolves around data.

These days everyone is talking about building data lakes for their machine learning initiatives. But you can't simply save all of your data to a lake and forget about it. Just as in sharing code and models, your data teams must collaborate closely on the preparation, use, and governance of data that is used for machine learning modeling.

Data lakes have been around for some time. But, "a few years back, Google and then Facebook said, 'there's value in the data, so we're just going to save everything," says Harrison. "A lot of companies have since adopted that same approach." However, over time some of these data lakes have become data swamps, where no one is managing the data, cleaning it, or tracking it over time. This can make it difficult for data scientists to extract useful data.

NOTE

To learn how to build and extract value from a data lake in the cloud and take advantage of the compute power and scalability of a cloud-native data platform, check out the O'Reilly report "Operationalizing the Data Lake".

Another potential problem with data is access rights—or, more accurately what happens when data scientists lack them. The most common scenario is that they must file tickets to have data engineers create and provide them with access to new datasets. Depending on the size of the raw data, the number of sources involved (and therefore the data needed), that could lead to significant delays in model development.

Enabling self-service access to data is one way to address this issue. It allows data scientists to access the data themselves without waiting for someone else to grant access or retrieve the data. At Qubole, for example, we provide self-serve access to data and store the metadata in the cloud so that any person with the appropriate privileges and access will be able to see the latest schemas as well as the metadata for all the different data sources. This avoids bottlenecks and, we believe, allows companies to innovate faster.

Enabling self-service access to data is critical when the goal is to efficiently build and productize machine learning models without incurring significant bottlenecks.

For example, Natali's data science team at Neustar tries to avoid depending on others to prepare data for them. "We like to start from the very beginning, with the raw data, to do the analyses, data cleansing, and transformations ourselves."

When his team is developing machine learning models, it's otherwise very difficult to understand how data has been transformed. He doesn't like having to guess or to ask people to recall, "what did you do here? What did it look like before you aggregated or changed it?"

There are other ways of tracking how data has been transformed. For example, rather than simply writing ETL scripts to pull data in and out of data lakes, a more evolved practice for sharing data is to build enterprise-grade data pipelines. These pipelines would automate connecting to data, pulling it in, doing the required transformation, and delivering it into the chosen environments, all with the necessary data-governance policies in place.

In fact, data governance considerations abound. For data scientists to build good models, they need to have deep insight into their data, what their data means, and the origin of their data. Sometimes, this can happen only by exploring the data for themselves while collaborating closely with data engineers.

Collaboration with Business Decision Makers and Executives

Explainability and auditability of machine learning predictions are increasingly important as the technology enters the business mainstream. Data scientists, therefore, must consider how to explain the results of their models to nontechnical or semi-technical colleagues or business executives. Planning how they can audit their models is also critical for compliance reasons.

This is challenging. Data scientists might have iterated the model multiple times, each time changing parameters, trading data, and adjusting different linear and cross-sector processes. How do you account for all of these iterations when considering audibility? And how do you know when you have developed the best possible model?

In the business world, explainability is often more important than accuracy. "Because in the commercial landscape you have to get buy-in from your customers," says H2O's Hall.

Hall says he's seen a lot of machine learning projects fail. "I would say the two reasons that they fail are, 1) because they're too complex or convoluted to be deployed, and 2) they're too complex or convoluted to be explained to either the customers, or the regulator, or the business partner," he says.

Many people say the value of machine learning hinges on whether it can be explained. For example, if you're an investment firm and you're using machine learning to predict what stock to buy or sell, being able to explain why your model recommends buying $50 million of stock is critical.

"I'm pretty sure the first question the traders are going to ask is, 'why?'" says Mehanna, formerly of Google Cloud. "That's just us as human beings, that's how we operate. We need to be able to trust that a system is reliable."

Over the past two years, tools to help people explain models have proliferated. But this is all very new territory, and there isn't necessarily one solution to point to.

One thing that is gaining in popularity is the notion of post hoc explainability. "That's where you go through the typical data science workflow, and then shoehorn explanations onto your model in the end," says Hall. "And if you're willing to work on it, you can do a pretty good job."

Complexity is a detriment to both deployment and explainability. So, making things only as complex as they need to be is also key for successful collaboration.

"We're all technologists, and we want to use the latest innovations," says Hall. But, unfortunately, a lot of times that devolves into *solutionism*—using a hammer just because you want to use the hammer. So, first and foremost: how is this overly complex model that you're making going to be used to make or save money, or provide value for the business in some other way?

Another challenge related to explainability is that too much emphasis is often placed on how models perform on static test data. Even if it's out-of-time test data—which hopefully it is—that's still a poor proxy to the actual business value of the model and how the model's going to perform on new live data streams.

It is therefore necessary to question how much time you spend fine tuning very complex models if all you have to work with is static test data. "You might be better off retraining a simpler model more often on new data," says Hall. If you work in a market where competitors are continuously entering and leaving, and new products are being introduced and retracted constantly, your data is constantly in flux. If you spend a long time fine tuning a model on a static snapshot of your market, it's going to be out of date relatively quickly.

Infosys is experimenting with explainability in the health-care and financial spaces, says Arora. In fact, building explainability into its products is a critical point for Infosys customers.

In summary, enterprises first need to organize their teams of data scientists, data engineers, machine learning engineers, and data analysts in a way that fosters collaboration. You need to establish a culture that encourages and rewards sharing. And, the complexity of models needs to be determined by the level of explainability

required, which in turn is determined by the business impact of a false prediction.

Problem 4: Building (or Renting) Adequate Machine Learning Infrastructure

Machine learning in the enterprise requires significant resources—human and nonhuman. When building an infrastructure capable of doing machine learning at scale, resource provisioning must take into account three factors: central processing units (CPUs), graphics processing units (GPUs), and storage.

The use of GPUs, in particular, has led to a significant speed-up of model training—so much so that some companies are even building hardware for machine learning that goes beyond traditional GPUs. Google designed a special type of hardware called a tensor processing unit (TPU) specifically for TensorFlow that is supposedly faster than the NVIDIA GPU. But the overwhelming majority of companies involved in machine learning buy or rent traditional GPUs.

Despite their value, managing the infrastructure with GPUs is an order of magnitude more complex than with CPUs. Moreover, the cost of managing on-premises GPU hardware has been increasing, with GPU card price tags reaching several 10s or 100s of thousands of dollars in some cases.

How you provision GPUs is critical. Drivers need to be installed before your machine learning library can communicate with GPUs. These are actually quite difficult to install and manage because most machine learning frameworks were not built with GPUs in mind. You also need to make sure that the GPUs will work with the particular machine learning library that you choose.

Several machine learning frameworks, such as Keras, TensorFlow, Theano, and PyTorch, support GPUs. Many don't. Depending on the data available and the use cases they will need to support, data scientists need to decide which libraries or frameworks to use to train their models on GPUs.

So, our fourth machine learning problem focuses considerably on GPUs, which have become popular for training machine learning models: how do you provision them, build the proper architecture,

select the appropriate framework, and manage them without breaking the bank?

To GPU or Not to GPU?

Virtually all data scientists point to the cloud when asked if they utilize GPUs. All the major cloud vendors will rent GPU space to you. Although the vendors take care of infrastructure and management of the resources, provisioning and using GPUs is still quite difficult. Also, there are not too many machine learning libraries that utilize GPUs for training.

Infosys rents GPUs in the cloud to speed up model training. Arora says that emerging architectures are becoming more suitable to the GPU instruction set as compared to using just CPUs. "It depends on the use case, and the amount of data, and the performance that you're looking for," says Arora.

In particular, deep learning models have a lot of parameters to train that involve multiple parallel computations. This is where GPUs excel. With the popularity and increasing adoption of deep learning, GPUs are also getting more popular.

But not all machine learning problems benefit from the use of GPUs. Harrison suggests asking yourself the following questions: Do you really need on-premises GPUs? Are you doing deep learning on unstructured data? Can you rent GPUs or do you need to have a local GPU farm? Do the cloud providers you are considering have support for the libraries that you want to use?

Harrison knows several enterprises experimenting with deep learning. They have large amounts of unstructured data, and they have their own GPU farms because, for them, that makes more sense than renting. But if you are smaller and don't have the IT budget or the engineering staff to handle such deployments, you might want to go to Software-as-a-Service (SaaS) machine learning providers or hosting providers that can take on that task.

"It depends on where your engineering is located, what libraries you have, and what kind of data you have," says Harrison. Those questions will drive what infrastructure you need.

But even having your own on-premises GPU farm might not be enough when you have high variability in machine learning work-

loads. This is where the scalability of the cloud proves to be a real differentiator.

"The beauty of the cloud is this concept of bursting, which companies can do when they get into a crunch in resources," says Mehanna. "I hear from companies that have a lot of GPUs on premises, but no matter what, their machine learning demands at some points in the year surpass their capacity. They love the idea that they can burst into the cloud and deploy those workloads on enormous amounts of accelerators and machines."

Cost Concerns

For companies that are sensitive to the cost of purchasing GPUs, says Hall, a solution might be cloud services. "With AWS Lambda, all you do is write code, and with Infrastructure-as-a-Service, or even Platform-as-a-Service, the service takes that code and decides for you how many nodes to allocate, what kind of CPUs or GPUs you need, and makes sure that it allocates those only for the period of time that you're going to need it," he says. It then de-allocates it as soon as it's done. "You're also making sure that you're only paying for what you use," says Hall.

Arora hasn't run into any trouble provisioning GPUs or installing GPU drivers, and cost isn't an issue because of the way that Infosys structures its costs. "Given that we work in a managed services construct, GPU cost is just a 'pass through' expense for us," says Arora. "We do have some solutions that we are building that will be more sensitive to the cost of tuning models on a GPU farm, but if the infrastructure is provided to the team as the platform is being installed, or a particular use case is being built out, it is the client's cost to absorb," he says.

Regulatory Constraints Around Data Location

In some cases—for example, health care, General Data Protection Regulation (GDPR)–related, or IoT use cases—it is not desirable to transfer data to the cloud or the on-premises datacenter to train models. These use cases might require special infrastructure planning.

Because no one size fits all when it comes to infrastructure, how do you know what kind of infrastructure you need? You need to talk to your engineering people, and you need to figure out how you're

going to deploy your model. "Again, this is where collaboration is super important. You especially want buy-in from engineering since, presumably, they're going to be the ones who are maintaining your systems and keeping them up," says Arora.

Problem 5: Scaling to Meet Machine Learning Requirements

Our fifth problem is scalability. Machine learning frameworks perform mathematical computations, which can become quite complex. Additionally, in every organization, you have scripts around the frameworks that automate the flow of data. As the data grows, this automation grows more challenging.

In fact, if you look at any software project, it's really about managing the flow of data from one point to the other, automating it, and determining how best to optimize each stage. With data science projects, you need to find a way to best use the time of the data science professionals involved to ensure that any model deployment minimizes errors. But, at the same time, it's also important to have standard processes in place to manage errors and mitigate risk when they do occur in production.

For example, you need to find a way to set up ETL jobs to clean, join, or prepare data that does not fit into a single machine. Even if you sample the data, you might want to test out tens or hundreds of models in parallel to save time. Additionally, you might want to train the model on the entire dataset in a distributed fashion. All this requires a focused effort on making infrastructure scalable.

Machine learning scalability requires planning. And it requires the ability to move quickly and rapidly process large volumes of data, to either build machine learning models or score new data against the machine learning models to make predictions, recommendations, or deliver machine learning–based insights. We believe that the elastic scalability that cloud-native platforms offer is the most cost-effective solution to process these big data workloads.

Another consideration is that you need to choose the best-suited framework for your needs. Be aware that the framework landscape is constantly changing. There are many different ones and many different ways to train your machine learning models. Thus, no one particular method is going to be the optimal solution forever.

From a data-science point of view, scalability issues can take various forms. Chief among them are the following four:

Being able to support data processing at scale
: Machine learning requires you to process huge amounts of data—too much, in fact, to fit onto a single machine. How do you do this, efficiently and cost effectively?

Executing hyperparameter optimization
: This involves running multiple experiments in parallel by tweaking parameters in the training algorithms. What's the best infrastructure to support this?

Performing distributed training
: This is where you can train the model in a distributed network across machines. We believe this is best done in the cloud.

Supporting growing numbers of users and applications
: Scalability means being able to support increasing numbers of users (data scientists, citizen data scientists, and analysts) while also supporting the growing numbers of applications that utilize data science.

None of these challenges have an easy solution. You need to plan for scalability up front in the tools that you use and the processes you set up.

Additionally, you need to keep in mind that data scientists are expensive resources; they (and their team leads) need to ensure that their time is well utilized. Not everything can be automated, but many manual tasks can be, which will allow your human resources to scale their efforts.

Data Processing at Scale

As previously mentioned, machine learning generates huge amounts of data that can't possibly fit onto one machine. Thus, you have no way to do joins or select statements on your tables. This is where big data engines—and the cloud—are both essential, because if you remain on-premises, using Hadoop or other big data technology usually isn't good enough to scale as high as you need to.

Of particular concern is getting access to large amounts of data so that you can pick the most relevant data for your machine learning project or experiment. So, being able to scale to do experimenting or

exploration of big data is essential, even if you are doing it only for preprocessing purposes.

"Most companies have lots of data, which may or may not be clean, or may or may not be relevant. You may not need all of the data for your particular use case," says Arora. "There are situations where the initial dataset is in terabytes, but actually when it comes down to a usable dataset it basically is just a couple gigabytes."

Usually Arora's team begins an engagement by going through a proof-of-value or proof-of-concept experimental phase. This is a quick four- to six-week period when a small portion of the sample data is shared to show the promise, or proof of value. Although he can usually do this on a laptop, when that phase is finished and he graduates to the entire dataset, the ability to scale is critical.

When it comes to the digital markets served by Neustar, datasets are pretty large, and to scale to his firm's needs, the cloud is essential. Typically, such data sets comprise tens of billions of rows with hundreds of columns. "We know each query will take a couple of hours, and that we're likely to run into difficulties," Natali says. "We need to have enough computation resources to handle it."

It's very common for his team to work on several queries at the same time, so that they're not just sitting in front of a computer screen waiting for a query to come back. His team is therefore always juggling resources, even in the cloud. It's important to be mindful of costs. "For a particular calculation, you could run with 10 nodes and take 100 minutes, or run with another node, it takes something like 10 minutes. You can find out that the cost will be roughly the same," he says.

Hyperparameter Optimization

When optimizing their models, data scientists often need to run multiple experiments, sometimes dozens, by tweaking parameters in the training algorithms and rerunning the models. This process is much more efficient if the experiments are done in parallel. It's called *hyperparameter optimization* because the parameters are the coefficients of the model. But a model, of course, has parameters itself. You use a type of regression called *elastic net* in which you can tweak a parameter to do shrinkage on the coefficient.

For example, you could have 10 or more different parameters that you want to test on five different algorithms. This comes to 50 different models to test. You can either run them in sequence on your laptop—which would take days—or train them in parallel using hyperparameter optimization via open source tools such as Spark, or through automated cloud services such as AWS SageMaker.

Using hyperparameter optimization, you take all 50 models, run them simultaneously using a machine learning engine, and get the results in an hour. You then choose the best one. An organization could either build and maintain the framework to support hyperparameter optimization, or it could use a cloud-native platform.

Distributed Training

Suppose that you want to train your model, but you don't want to do so on just a sample of your data: you want to train your model on your entire dataset. But only a few libraries support distributed training. This is one of the top reasons data scientists are forced to sample data. If they have a cutting-edge algorithm that they'd like to use, it's probably not available in Spark and Hadoop.

However, some distributed computing frameworks such as Spark MLib do allow distributed training, wherein you can train the model in a distributed network across machines. Although you can do this on-premises, it is much more effective, and cheaper, if you do it in the cloud.

Arora frequently models his data using a distributed training framework such as Spark ML, H2O Sparkling Water, or XGBoost. His platform is cloud based, and he uses a number of managed sources. He uses a Spark engine to do distributed training.

Neustar's Natali says it's common for his data team to use a sampling strategy to increase the ratio of successes to failures. But he doesn't do so for size reasons; for example, so he can train on a single machine. That doesn't work for his business. "We prefer to train on a distributed system or a system that is beefy enough to handle a large amount of data," he says. That means the cloud.

Precisely because of scalability, going with open source solutions is quite important to Neustar. Natali thinks it's one of the most important aspects of choosing a tool. "Data science people used to depend on MATLAB, which is still quite heavily used in engineering fields,"

he says. "It does have some nice features and some good packages that people can use, and everything's curated. So there are advantages." But after you move away from using something like MATLAB on your own computer to scaling it in a distributed system, like a cluster in the cloud, the licensing becomes prohibitive. Whereas with an open source product, you never need worry whether you are using 100 nodes or scaling to 1,000 nodes.

Supporting Growing Numbers of Users and Applications

Rather than volume, the real dimension of data that's the most difficult to deal with is velocity—the attribute that measures how often and how quickly the data changes. You might be looking at HR analytics for which your profile information is pretty static and doesn't change for weeks and weeks. Or you might be looking at telematics, for which information is changing on a second or maybe even subsecond basis. "Velocity is really the dimension that you have to look out for," says Overton.

Variety is also a factor when it comes to scalability. There are cases for which you might be taking location information and combining it with weather information and patient history information, and taking otherwise disparate data sources and combining them.

Why Automation Is Key for Scalability

Many of the challenges of scalability come down to finding efficiencies, and that leads inevitably to questions of how you can automate processes. Data scientists do many repeatable, mundane things in their work that can be productized and allow them to scale their time. But there are obviously some things that you can't automate. This is the "art" aspect of data science.

Overton recommends buying, not building, automation tools. "Building them sounds great when you first do it, but what ends up happening pretty quickly is you spend more time and resources on maintaining these automation tools than you would have if you had just bought them," he says.

Qubole's cloud-native data platform, for example, enables organizations to automate complex data-processing tasks, resulting in faster time to value and lower infrastructure costs. Unlike legacy on-premises platforms, Qubole provides workload-aware autoscaling,

automatic cluster start/stop, and heterogeneous cluster configurations to optimize and reduce infrastructure costs.

Problem 6: Enabling Smooth Deployment of Machine Learning Projects

Problem number six is the culmination of the five previous problems, which is how to deploy machine learning products at scale. The specific issues that you'll run into come down to what use cases you're going to support and what frameworks you're going to use. The sad fact is that quite a few machine learning projects do not succeed. Here are some of the chief roadblocks and what enterprises are doing about them.

Batch versus Real-Time and Deploying to the Edge

To understand these issues, you must first distinguish between running the process on a regular basis (batch) or whether it's continuously up (real time) in a customer-facing application. A batch deployment runs on a schedule—weekly, nightly, hourly—whereas a continuous deployment requires an immediate response with a prediction.

Another consideration is deploying models to the *edge*. In this deployment scenario, the edge could be anything. The edge could be a datacenter sitting in a remote industrial site, or the edge could be a small device sitting in a car. It might not always have full connectivity. So how do you ensure that your model is able to correct itself when it encounters changes to the data, and how does it communicate those changes, and how do you optimize the model for deployment on smaller devices? The challenge of doing machine learning on the edge is managing to get insights fast enough to be useful. For example, a consumer could be getting off a train and pass by their favorite store. If a vendor can detect that in time, it can use a machine learning model that's deployed on the consumer's mobile phone (an edge device) to deliver a promotional offer based on the consumer's preferences and propensity to buy.

"Getting a model out to the edge is fairly straightforward," says Overton. "But putting something out on the edge that's going to be intelligent enough and fast enough to give you value—that's something else."

Experimentation versus Production

When you're building a laptop model and you don't need to deploy it yourself, problems one through five are not that onerous. It's when you're trying to deploy a model in a production setting—meaning train or score the model on a recurring basis in a batch job, or set up a machine learning product inside of your application—that things get tricky.

At this point, many machine learning projects fail. And the reason is because the models are overly complex or utilize tools or libraries that break in production. "If your model is too complex, then you may never be able to deploy it," says Hall.

The good news is that more and more tools—including H2O's tools—are beginning to make representations of the models they train in languages such as C++ and Java. "So I can train an H2O model in R or Python, using R or Python code from a Jupyter Notebook, or RStudio," says Hall. "And then, when that process is over, I get a piece of Java code."

To be fair, Hall says, H2O isn't the only company doing this. When choosing a tool, it's important to break models into two buckets. Going into the first bucket are internal self-service type applications. "So I'm a data scientist, and I'm making something for a business analyst inside my company," he says. "There, Docker, R, Python, Jupyter Notebooks, and other tools are fine."

But the other bucket (in which that approach breaks down) is for public-facing, mission-critical, real-time tasks, says Hall. For example, when you're deciding whether a person is approved for the credit card or not. Or you are deciding what coupon to send the person in real time. "Or you're deciding how large an insurance policy should be," he says.

The mission-critical applications typically still require some kind of translation between that R and Python tool stack into C++ or Java. "And that is really hard if you don't have a tool that does that for you automatically," he says.

If the environment changes from that snapshot, the model will perform poorly. "Which means you are losing money or losing value in some other way," says Hall. And this idea of model management, model monitoring, and complex deployment strategies is very important.

"I'm starting to see more and more vendors come alive in this side of the market, and customers to be increasingly aware of this," he says. But for a long time, people would deploy models and leave them out for years, not having any idea of whether the model was making money, losing money, or behaving terribly. "So monitoring the model once it's been deployed is just hugely important," he says. "Especially when you move away from linear models."

Setting expectations of the machine learning project is also important.

Before models go into production at Infosys, there are certain criteria that must be met, which the firm's data team agrees upon in advance. For example, production models must come with a certain level of explainability, visibility, and traceability. These are points discussed and agreed upon as part of the proof of concept.

"It's difficult to say, 'Here's a checklist,' but you need to keep in mind as you go from experimentation to production that there is a method to the madness," says Arora.

Continued Auditing Is Critical

After the model is in production, you must continue to audit it.

"We have situations where the basis for a deal is that we will continue to provide a certain level of accuracy, or a certain level of savings because of a particular model," Arora says. "So model drift is always being measured because you need to keep that SLA in place."

According to Overton, the real problem with deployment for enterprise-scale systems is being able to understand the level of risk that you are incurring from the model learning things that you hadn't intended. "This lurches over into ethics, and compliance, and security with AI and data science," he says.

Overton compares machine learning to straight computer programming. "I started my career as a programmer," he says. "And the mindset when you're a programmer is that you want to get this thing to do exactly—I mean exactly—what you specified it to do, and if it veers from that even a little bit, that's a bug and you need to fix it."

As a data scientist, on the other hand, you're creating general algorithms, and what you want them to do is to learn and to produce

behavior that you hadn't anticipated. "So after you put something into production, you actually have a product that's getting new features on its own," he says. One of the key challenges with a system that can learn is this: what is it learning when it's in deployment?

"You train it on a certain corpus of data, and you see the behavior and you think, 'okay this is great, I'm going to put this into production,' but when you put it out into production, it could be exposed to an entirely different set of information, and that's when your deployment puts you at risk," says Overton.

Thus, in his mind, the biggest challenge in deployment is being able to detect the difference between the data your model is being exposed to when it's out in deployment and what it was tested on.

How do you do that? "Profile your AI," says Overton. "Make explicit your understanding of how it's going to behave and the boundary conditions under which this thing is likely to start acting crazy. And then you monitor it and make sure that it doesn't freak out."

What's interesting is that the tools for profiling your AI models are themselves AI. So, you end up using machine learning algorithms to determine what factors your algorithm is using to drive decisions.

At the travel conglomerate, Downes says that there is so much concern about bias creeping into machine learning models that the data team collaborates with business analysts to perform what they called "end-to-end scaffolding tests." In this scenario, the data scientists build the model, demonstrate it, and put it onto the live site but without sending any traffic to the model. "Our content specialists at that point will look at it, tweak it and poke it, and see if they like it."

This experimentation is critical because you need to ensure that you're not injecting biases that are socially irresponsible or that you could even potentially be criminally liable for. "For big, heavy-use day-to-day applications that use machine learning, that's essential," says Downes, who is also a big advocate for "ground truthing" data. This involves injecting synthetic data into the system and looking at the output at various stages to see whether it's what you expected. "You could do this at scale by setting up automated tests, and it would be extremely useful," says Downes.

Conclusion

You can solve each of these six problems individually with a DIY approach. However, it is preferable to use a platform that is *built* for enterprise data science. A cloud data platform provides the best possibility of success by addressing all of these challenges.

You can solve the first machine learning problem we identified, that of disparate interfaces, by using a cloud-native data platform that preconfigures engines and interfaces to interact smoothly with whatever tool your data scientists choose.

The second challenge, that of reconciling different environments, can best be addressed with a platform for data science that is able to support any and all available frameworks. Again, that points directly to a cloud-native data platform.

But interfaces, environments, and machine learning models don't work by themselves. In the third challenge, we talked about how data science is a team sport. Having cloud-native tools that allow continuous collaboration and governed searches of the code, data, and metadata of peers can lead to more optimal outcomes.

Our fourth machine learning problem focused on machine learning infrastructure; specifically, GPUs. We discussed the difficulties in provisioning them, building the proper architecture, selecting the best-suited framework, and managing them without breaking the bank—or your backs.

Our fifth problem was scalability. Deployment of enterprise-wide machine learning solutions requires the applications to rapidly scale to accommodate variability in usage or data. A cloud-native data platform provides the agility to scale up compute capacity to meet demand and scale down when the usage drops. In addition, the consumption-based pricing of most cloud platforms ensures that you pay only for what you use. Thus, we believe cloud platforms are the best solution to the challenge of scalability.

Problem number six is the culmination of the five previous problems, which is how to deploy machine learning products at scale. Cloud-native data platforms, like Qubole, offer such scalability. If a particular workload needs additional compute power, the cloud infrastructure can easily expand to meet those needs.

About the Authors

Piero Cinquegrana is an accomplished data scientist. He worked for Qubole as the senior product manager for Qubole's data science offering.

Prior to Qubole, Piero was a Senior Data Scientist for over five years at Marketshare, a leading marketing analytics firm in Los Angeles. In that role, Piero was a key contributor in several of Marketshare's products built to measure and optimally allocate marketing dollars across a variety of industries. (Marketshare was later acquired by Neustar in 2015.)

Matheen Raza is a Machine Learning enthusiast. He worked for Qubole as a Product Marketing Manager for data science and machine learning. Prior to Qubole, Matheen was at Infosys, where he was a Product Marketing Manager for Infosys Nia—Infosys's AI/ML and Big Data Platform. Prior to that, he was at Intel where he held multiple roles across Engineering and Product Marketing.

Matheen holds a B.S. in Electrical and Electronics Engineering from the University of Madras, a Masters in Electrical and Computer Engineering from Colorado State University, and an MBA from the University of California, Berkeley. Outside of work, Matheen enjoys basketball, snowboarding, hiking, and spending time with his wife and two young boys.